PENJELASAN DARI KODE PEMBELAJARAN
PERTEMUAN 6

Nama : Muuhammad Luqmqnul Fikri

NIM : 231011400546

Kelas : 05TPLE017

 Struktur dan Tujuan Umum

Program ini:

1. Memuat dataset kelulusan mahasiswa (processed_kelulusan.csv).
2. Jika file tidak ada → buat data dummy otomatis.
3. Bagi data menjadi train–validation–test (70–15–15).
4. Bangun pipeline Random Forest dengan preprocessing lengkap.
5. Lakukan validasi silang + tuning hyperparameter (GridSearch).
6. Evaluasi model dengan F1 score, ROC, confusion matrix.
7. Simpan model (rf_model.pkl) dan beberapa hasil gambar.
8. Coba prediksi data sample baru.

◈ 1. Import dan Setup

```
# ===== tugas6_fixed.py – Random Forest untuk Klasifikasi (robust untuk dataset ke
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')  # Suppress warnings untuk clean output
```

Import pustaka yang dibutuhkan: numpy, pandas, matplotlib, sklearn.

warnings.filterwarnings('ignore') → sembunyikan warning agar output bersih.

🔷 2. Muat Dataset atau Buat Dummy

```
csv_file = "processed_kelulusan.csv"
if os.path.exists(csv_file):
    df = pd.read_csv(csv_file)
else:
    # buat dummy data kalau file tidak ada
```

Jika file processed_kelulusan.csv ada → dibaca.

Kalau tidak ada → program buat dataset dummy berisi kolom:

- IPK
- Jumlah_Absensi
- Waktu_Belajar_Jam
- Rasio_Absensi
- IPK_x_Study
- Lulus (label target: 1 = lulus, 0 = tidak lulus)

Dummy disimpan agar tidak perlu dibuat ulang nanti.

🔷 3. Pemisahan Data Train, Validation, Test

```
X = df.drop("Lulus", axis=1)
y = df["Lulus"]
```

X = fitur input, y = label target.

```
# split 70/15/15 — stratify sekali + cari seed agar test punya 2 kelas
X_train, X_temp, y_train, y_temp = train_test_split(
    X, y, test_size=0.30, stratify=y, random_state=42
)
```

Data dibagi 70% train, 30% sementara (val+test) secara stratifikasi.

Lalu kode mencari seed (random_state) yang membuat test set tetap memiliki 2 kelas (0 dan 1):

```
seed_found = None
for rs in range(500):
    X_val_try, X_test_try, y_val_try, y_test_try = train_test_split(
        X_temp, y_temp, test_size=0.50, random_state=rs
    )
```

Ini penting agar model bisa evaluasi ROC dengan benar (butuh dua kelas).

Seed terbaik disimpan ke seed.txt.

◈ 4. Preprocessing dan Pipeline

Menentukan kolom numerik dan kategorikal:

```
# ---------- Langkah 2: Pipeline & Baseline RF ----------
num_cols = X_train.select_dtypes(include="number").columns
cat_cols = X_train.select_dtypes(include="object").columns  # Handle kategorikal jika ada
```

Lalu buat ColumnTransformer:

Numerik → di-impute median lalu distandarisasi (StandardScaler).

Kategorikal (jika ada) → impute “missing” dan OneHotEncoder.

Semua dikombinasikan dalam satu pipeline pre.

◈ 5. Model Baseline Random Forest

```
rf = RandomForestClassifier(
    n_estimators=300, max_features="sqrt",
    class_weight="balanced", random_state=42
)

pipe = Pipeline([("pre", pre), ("clf", rf)])
pipe.fit(X_train, y_train)
```

Gunakan 300 pohon (trees).

class_weight="balanced" agar memperbaiki ketidakseimbangan kelas.

Semua preprocessing + model disatukan dalam pipeline.

Lalu model dilatih dan dicek performa awal di validation set:

```
pipe.fit(X_train, y_train)

y_val_pred = pipe.predict(X_val)
baseline_f1 = f1_score(y_val, y_val_pred, average="macro")
```

🔷 6. Validasi Silang (Cross-Validation)

```
# ---------- Langkah 3: Validasi Silang (pakai 2-fold biar aman) ----------
skf = StratifiedKFold(n_splits
cv_scores = cross_val_score(pi
```

(variable) cv_scores: ndarray

Validasi silang 2-fold untuk memastikan model stabil walau data kecil.

Mengukur rata-rata F1 score pada data latih.

🔷 7. Grid Search Tuning

```
param = {
    "clf__max_depth": [None, 12, 20, 30],
    "clf__min_samples_split": [2, 5, 10],
    "clf__min_samples_leaf": [1, 2, 4]  # Tambah untuk
}
```

Mencoba berbagai kombinasi parameter RF agar hasil optimal tanpa overfitting.

Evaluasi pakai F1 macro.

Simpan model terbaik (best_model).

🔷 8. Evaluasi Akhir di Test Set

```
y_test_pred = final_model.predict(X_test)
test_f1 = f1_score(y_test, y_test_pred, average="macro")
```

Mengukur performa akhir di test.

Cek overfitting:

Hitung F1 train vs test.

Jika gap besar (> 0.1) → tanda model terlalu menghafal data.

◈ 9. Confusion Matrix dan Grafik

```
plot_cm(cm, classes=["Tidak Lulus (0)", "Lulus (1)"], title="Confusion Matrix (test)", filename="cm_test.png")
```

Simpan visualisasi confusion matrix (cm_test.png).

Lalu buat ROC curve dan Precision-Recall curve, disimpan sebagai gambar.

◈ 10. Feature Importance

```
importances = final_model.named_steps["clf"].feature_importances_
feat_names = final_model.named_steps["pre"].get_feature_names_out()
# Bersihkan prefix "num__" atau "cat__" dengan aman
```

Menampilkan fitur mana yang paling berpengaruh terhadap hasil prediksi.

Menunjukkan Top 10 Feature Importan

◈ 11. Simpan Model

```
# ---------- Langkah 7: Simpan Model ----------
joblib.dump(final_model, "rf_model.pkl")
print("\n💾 Model disimpan sebagai rf_model.pkl")
```

Menyimpan model agar bisa dipakai ulang tanpa melatih lagi.

◈ 12. Contoh Prediksi Satu Data Baru

```
sample_data = {...}
sample = pd.DataFrame([sample_data])
pred = int(final_model.predict(sample)[0])
proba = final_model.predict_proba(sample)[:,1][0]
```

Coba prediksi data baru dengan nilai IPK, absensi, dll.

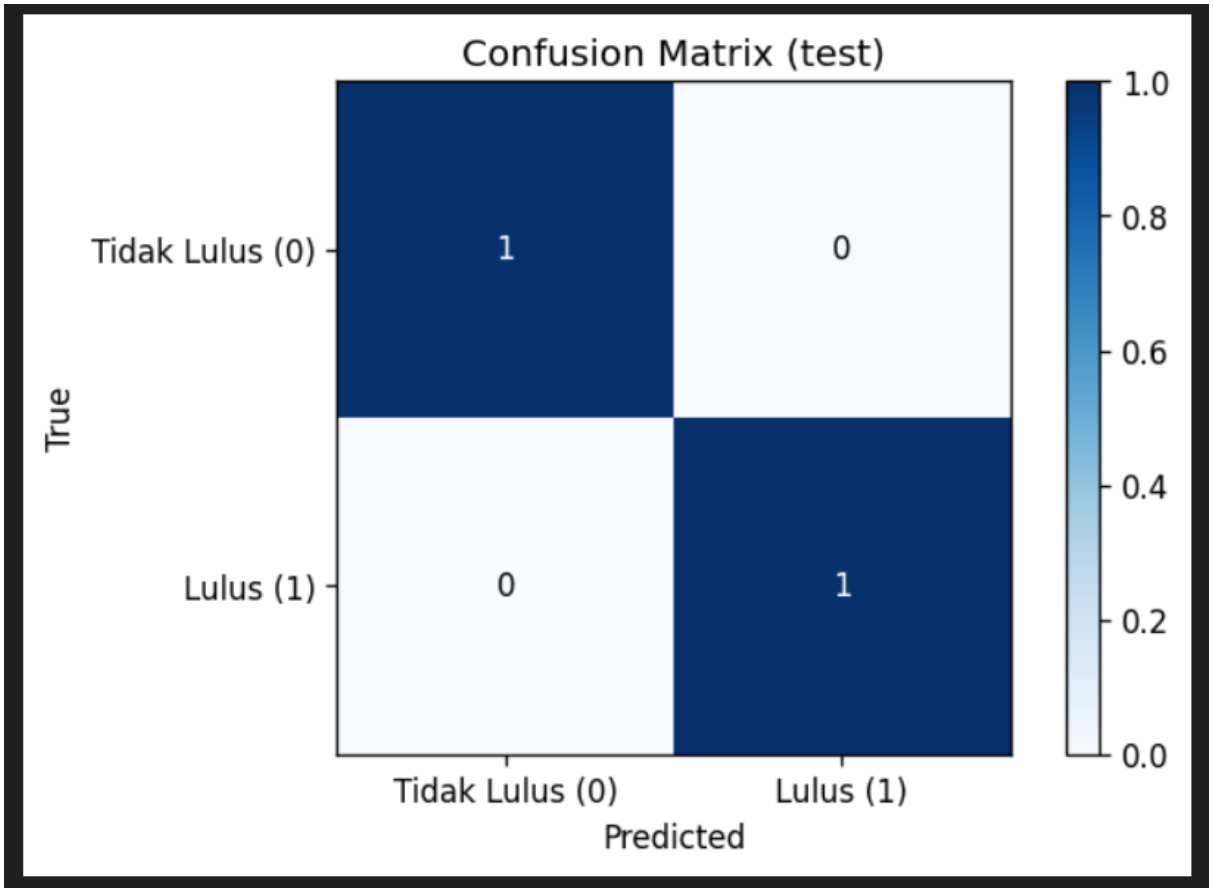
Menampilkan hasil prediksi dan probabilitas “lulus”.

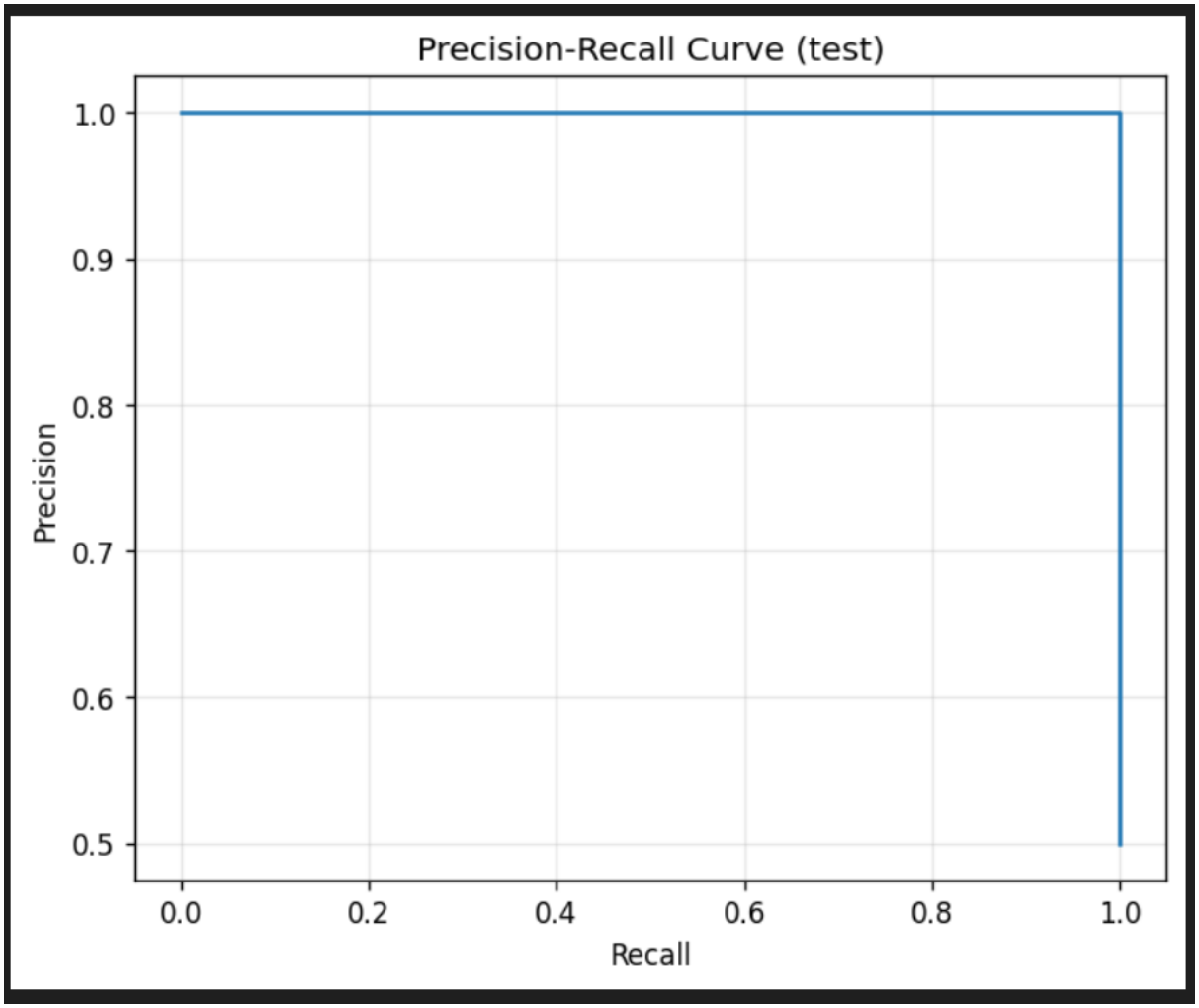
✅ Kesimpulan

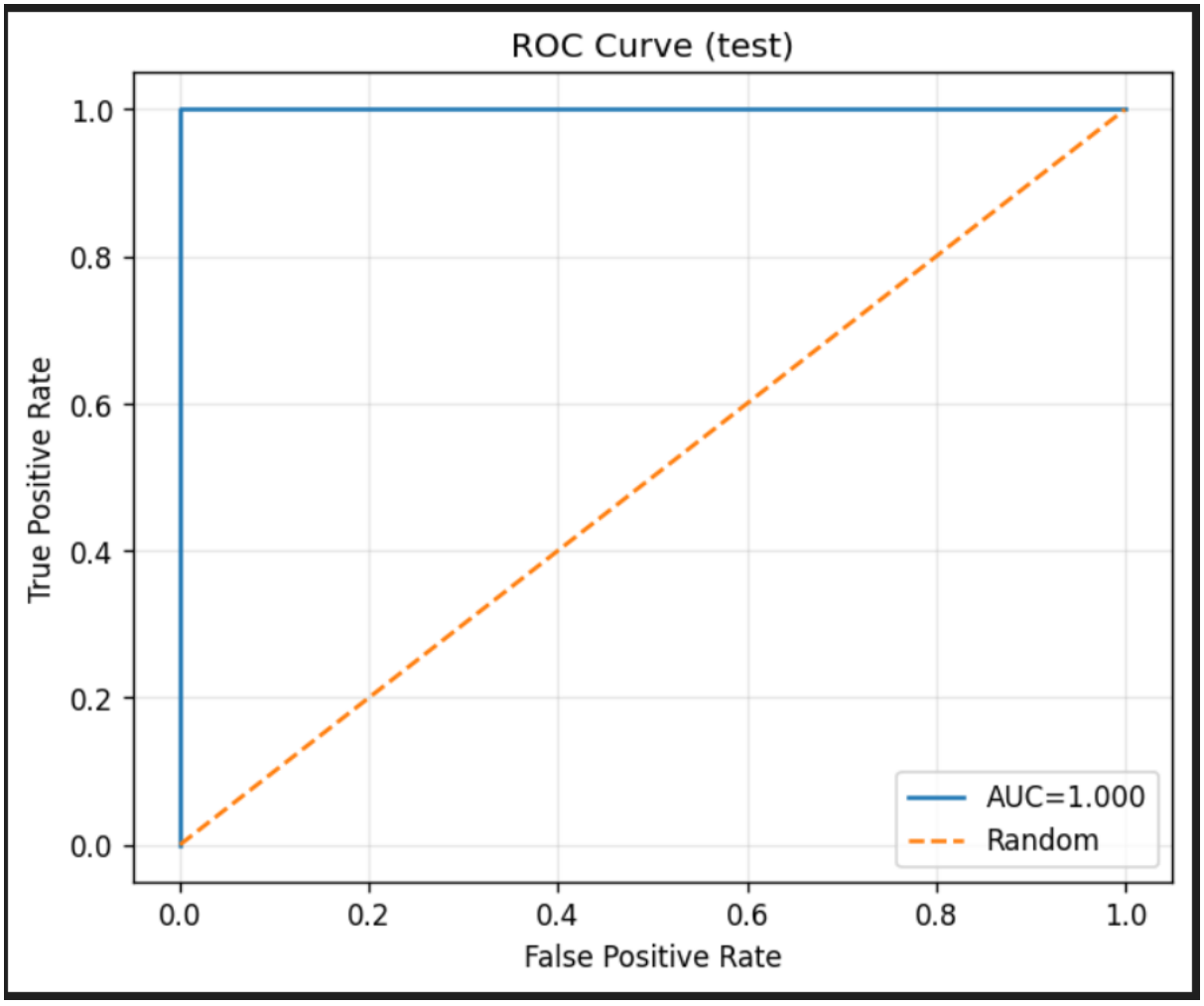
- Program ini adalah pipeline machine learning yang lengkap:
- Otomatis membuat data dummy jika file tidak ada,
- Memastikan data terbagi adil antar kelas,
- Menjalankan preprocessing otomatis,
- Melatih dan tuning model Random Forest,
- Mengevaluasi performa dengan berbagai metrik,
- Menyimpan hasil dan model siap pakai,
- Memberikan contoh prediksi akhir.

HASILNYA

```
✅ Loaded processed_kelulusan.csv (10 rows).
✅ random_state kedua = 0 → test set mengandung 0 & 1.
💾 Seed disimpan ke seed.txt.
Shapes: (7, 5) (1, 5) (2, 5)
Label count — train:
 Lulus
1    4
0    3
Name: count, dtype: int64
Label count — val:
 Lulus
0    1
Name: count, dtype: int64
Label count — test:
 Lulus
1    1
0    1
Name: count, dtype: int64

Baseline RF — F1(val): 1.0
              precision    recall  f1-score   support

           0      1.000     1.000     1.000         1

    accuracy                         1.000         1
...
💾 Model disimpan sebagai rf_model.pkl
Contoh prediksi sample → 1 (Lulus: Ya) | proba lulus: 0.983

✅ Kode selesai! Jalankan dengan data real untuk hasil akurat.
Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...
```


Confusion Matrix (test)
Tidak Lulus (0)
Lulus (1)
True
1
0
0
1
Tidak Lulus (0)
Lulus (1)
Predicted
1.0
0.8
0.6
0.4
0.2
0.0


Precision-Recall Curve (test)
Precision
Recall
1.0
0.9
0.8
0.7
0.6
0.5
0.0
0.2
0.4
0.6
0.8
1.0


ROC Curve (test)
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
AUC=1.000
Random